CS63 Spring 2019
Heartbeat Abnormality Classification

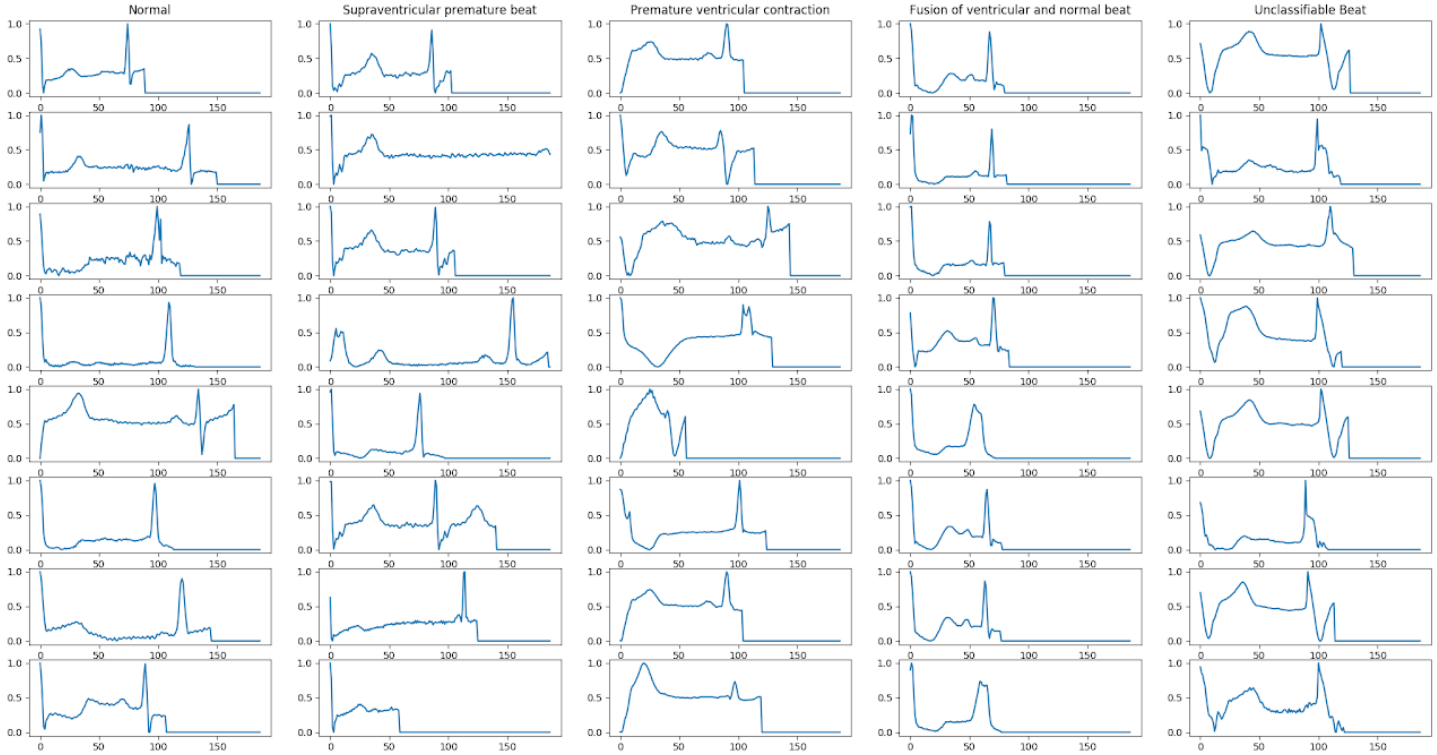
Holden Bridge & Rose Ridder

Introduction

The Problem

The heart is the most used muscle in the human body. Irregularities in its rhythm can reveal critical information about a person's health and well-being including immediate health characteristics such as stress, restedness, caffeine or alcohol consumption, electrolyte imbalances, as well as more long-term afflictions like heart disease, past heart surgery, heart muscle changes, and others.

Premature Ventricular Contraction is the most common heartbeat irregularity, which causes a stronger second beat. This is what typically occurs under stress or sudden fear, often described as the heart "skipping a beat" or "fluttering". Another particularly common irregularity is a Supraventricular Premature Beat, which is largely asymptomatic but an abundance of these beats may indicate more severe arrhythmias or other heart issues. These two, plus a fusion class of Normal and Ventricular irregularity, and an unclassifiable beat set of data were compiled in the MIT Arrhythmia Database and used in this project, examples of which are pictured below.



These 8 examples of each type of heart beat (Normal (N), Supraventricular Premature Beats (S), Premature Ventricular Contractions (V), Fusion Ventricular and Normal (F), and Unclassifiable (Q)) were included in the dataset and used in our testing data.

Identification

Identifying irregularities early and reliably can have measurable impacts on health outcomes. Early detection of these heart abnormalities can alert medical professionals to changes in a patient's status and allow issues to be addressed more efficiently. While doctors can detect some inconsistencies in heartbeats, particularly timing irregularities between the maximum electric potential change of a series of heartbeats, it is not an easy task. Classifying heartbeat inconsistencies more specifically is often done through painstaking effort and measurement of the QRS complex pattern, which includes the surrounding electrical potential changes rather than the single characteristic peak.

This database examines individual heart beats to classify normal and abnormal rhythms. This is particularly poignant because some of these disorders may be expressed only in a few beats at a time in the early stages of a patient's condition. Premature Ventricular Contractions (PVCs), for example, happen in most people at least once per day. This is not abnormal. However, more frequent PVCs can be a reason for concern. Detecting a statistically significant increase in PVCs can therefore help to alert medical professionals of an underlying heart condition.

By looking at the data, we see some patterns emerge particularly in the width of the peaks and the baseline of the data. The higher frequency deviations surrounding the peaks also vary in intensity. It is particularly these characteristics that a doctor would examine to identify beats as regular or irregular.

For this project, we developed a neural network to classify these heartbeat abnormalities. This involved examining and manipulating the data into neural network inputs and outputs, developing a neural network, testing and validating the network, and developing further assessment mechanisms, This could be used in hospitals alongside digital monitoring devices to immediately identify abnormal heart beats and alert a doctor if there is a statistically significant number of irregularities.

Method and Details

Finding and Interpreting the Dataset:

The first thing we did was observe how the data was structured on kaggle. The data set was derived from a popular dataset in heartbeat classification called the MIT-BIH Arrhythmia Dataset. The original training set had 87,000 examples and the test set had 22,000 examples. Each example measured a single heartbeat and contained 188 columns each with a numerical value. Each heartbeat could fall into one of the 5 categories mentioned above (N,S,V,F,Q). If the example did not need 188 columns it filled the remaining slots with 0, this is commonly referred to as zero padding.

Choosing a Neural Network:

After analyzing the structure of the data we decided that it would be best to use a 1D convolutional neural network with pooling. This allows the machine to look at a window of values and recognize features that may exists in the data. We thought that this made the most sense to use for observing heartbeat patterns. Another option we considered was a recurrent neural network, however this option is more complex and in order to achieve the same level of accuracy would require a much longer training time compared to the convolutional neural network.

Manipulating The Data For Our Network:

The next step in our process was shaping the data for our network. The data was already split into training and testing sets from kaggle so we did not need to do that. It was also normalized, so that the electrical potential of the beats was between 0 and 1. We did reshape the given data to be 3-Dimensional so that we could provide the first convolutional layer with a proper input shape. One thing we noticed early on was that there were noticeably more examples that resulted in a 0 (Normal Beat) than any other classification. We determined that this would be problematic because our network could classify everything as a 0 (Normal Beat) and still have a high accuracy due to the sheer number of examples corresponding to a Normal Beat. In order to avoid this issue we ignored the first 67,472 examples in the training set and the first 16,499 from the test set, all of which were classified as a 0 (Normal Beat). After removing these examples, our data set became much more evenly distributed and we felt that we were ready to move onto testing different aspects of network.

Designing Our Neural Network:

Since there are many similarities between the two datasets we designed our network very similarlly to the 2D convolutional network we built in Lab08. We converted our target layers to categorical vectors that utilized one-hot encoding. Making the target vectors this way allowed us to create a sequential neural network that ends with dense layer of 5 which uses the softmax activation function using Keras. When compiling our model, we used the Adagrad optimizer, categorical cross entropy loss function, and categorical accuracy to measure our model's accuracy.

Maximizing Performance:

The first thing we decided to add was multiple dense layers. Deep learning feed-forward neural networks are highly effective and we wanted to utilize this fact. We tested different sizes for these dense layers and different numbers of dense layers. Ultimately we decided to use 2 dense layers of size 50. We choose this because it would not add too many parameters and significantly improved the accuracy of the model.

Next we worked on implementing 1-dimensional convolution layers as well as pooling layers into our network. For the convolutional layer we tried using 16 filters and 32 filters, and we determined that the network performed better when using 32 filters. For the kernel size we

tested a wide range of values including 2,5,10,15, and 25. We saw a significant increase in performance from 2 to 5 but all values greater than 5 did not seem to significantly improve accuracy so we decided to use a kernel size of 5 for all of our convolutional layers. Using the same process we decided on using a pool size of 4 for all pooling layers. We tested many combinations of 1D convolutional and pooling layers with the goal of maximizing accuracy. These tests can be seen in the results section.

We also varied the number of epochs when training the data. We wanted to choose a number that avoided overfitting, did not take an excessive amount of time, and had a high accuracy when testing. These tests can also be found in the results section.

Results

Testing Different Numbers of Epochs:

Model used for testing variations of epochs: (73,799 total parameters)

```
neural_net.add(Conv1D(32, 5 ,activation="relu",input_shape=(187, 1)))
neural_net.add(MaxPooling1D(pool_size = 4))
neural_net.add(Conv1D(32, 5 ,activation="relu"))
neural_net.add(Flatten())
neural_net.add(Dense(50, activation='relu'))
neural_net.add(Dense(50, activation='relu'))
neural_net.add(Dense(5, activation='softmax'))
```

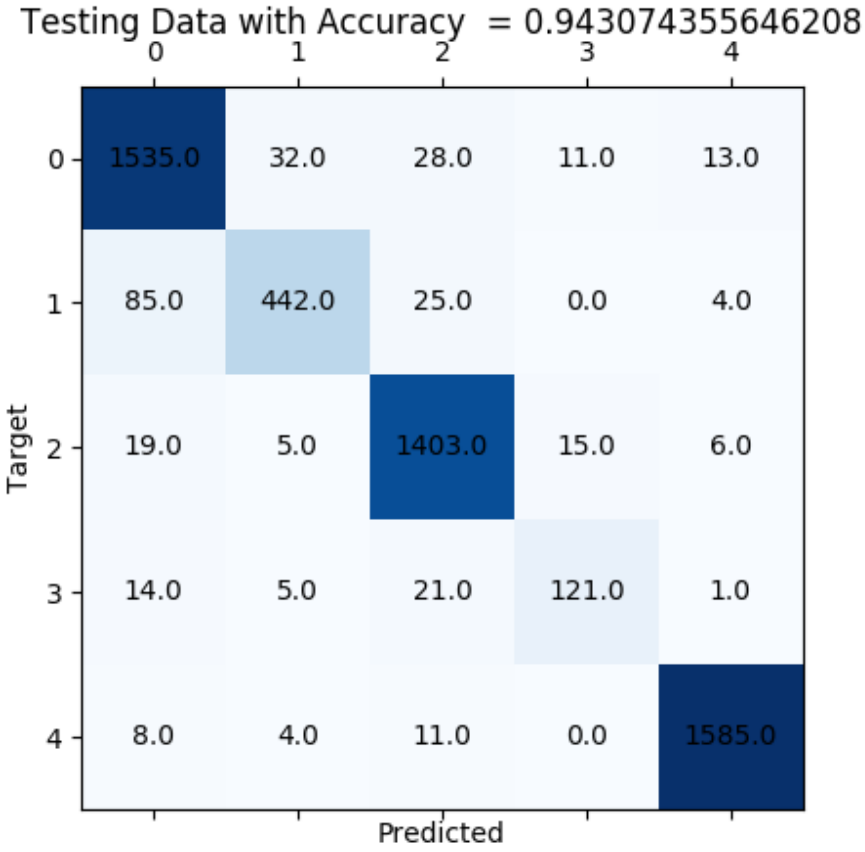
Each epoch took approximately 3 seconds to complete. The following table shows the accuracy on both the testing and training data for different numbers of epochs.

Number of Epochs	Training Accuracy	Testing Accuracy
5 epochs	92.58042027686486%	91.52605229000557%
20 epochs	95.38392590379445%	93.93658446133877%
50 epochs	96.21551638283039%	94.3074355646208%
100 epochs	98.00816651727916%	94.86371221954386%
150 epochs	98.8148590777811%	95.03059521602076%

After analyzing these values we decided that 50 epochs was an appropriate number. We considered how long it took the network to train, which was between 2-3 minutes in this case. We also wanted to avoid overfitting on the training data. We felt that the slight increase in testing

accuracy when using 100 or more epochs was not worth the risk of overfitting or additional time. In conclusion, we felt that 50 epochs had the best balance of time needed to train, avoiding overfitting, and having a high testing accuracy.

Confusion Matrix on Test Data for 50 Epochs



The confusion matrices on the test data for the other variations of number of epochs can be found in the appendix section.

Testing Different Numbers of Convolution and Pooling Layers:

When testing the different number of convolution and pooling layers we used 50 epochs each time because that was the number we selected after doing our epoch testing.

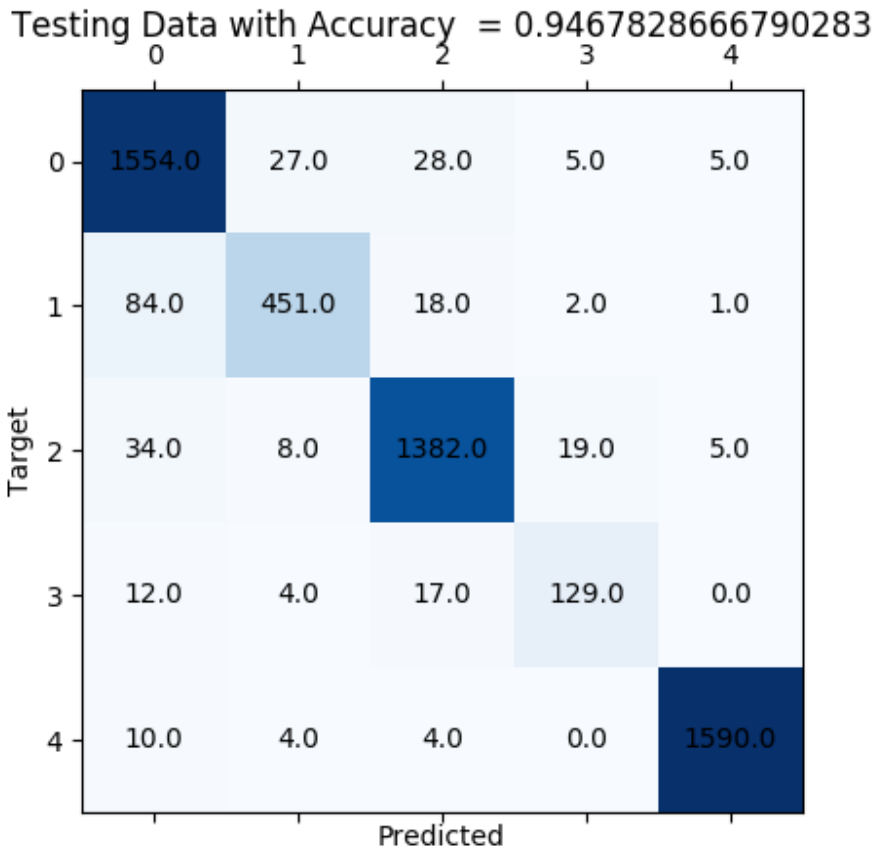
- All convolutional layers had 32 filters and a kernel size of 5
- All pooling layers had a pool size of 4

The following table shows the accuracy on both the testing and training data, the time needed for each epoch, and the number of parameters for the different combinations of convolution and pooling layers.

Layer Type	Time per Epoch	Number of parameters	Training Accuracy	Testing Accuracy
1 Convolution Pooling 1 Convolution	3 sec	73,799	96.9375560203167%	94.67828666790284%
2 Convolutions Pooling 2 Convolutions	4 sec	76,103	98.78498157554029%	94.34452067494901%
2 Convolutions Pooling 2 Convolutions Pooling	4 sec	32,903	97.69943232745743%	94.25180789912851%
4 Convolutions Pooling 4 Convolutions	7 sec	80,711	99.02400159346678%	93.43593547190802%

After analyzing these values we decided that 1 Convolution Pooling 1 Convolution is the best combination for our network. Increasing the number of convolutional layers increased the training accuracy but actually lowered the testing accuracy, indicating that the model was likely overfitting the training data. We also noticed that ending with a pooling layer significantly lowered the number of parameters but did not have much of an effect on any other aspects. In conclusion, we felt that 1 Convolution -> Pooling -> 1 Convolution was the best model for avoiding overfitting and successfully generalizing to the test data.

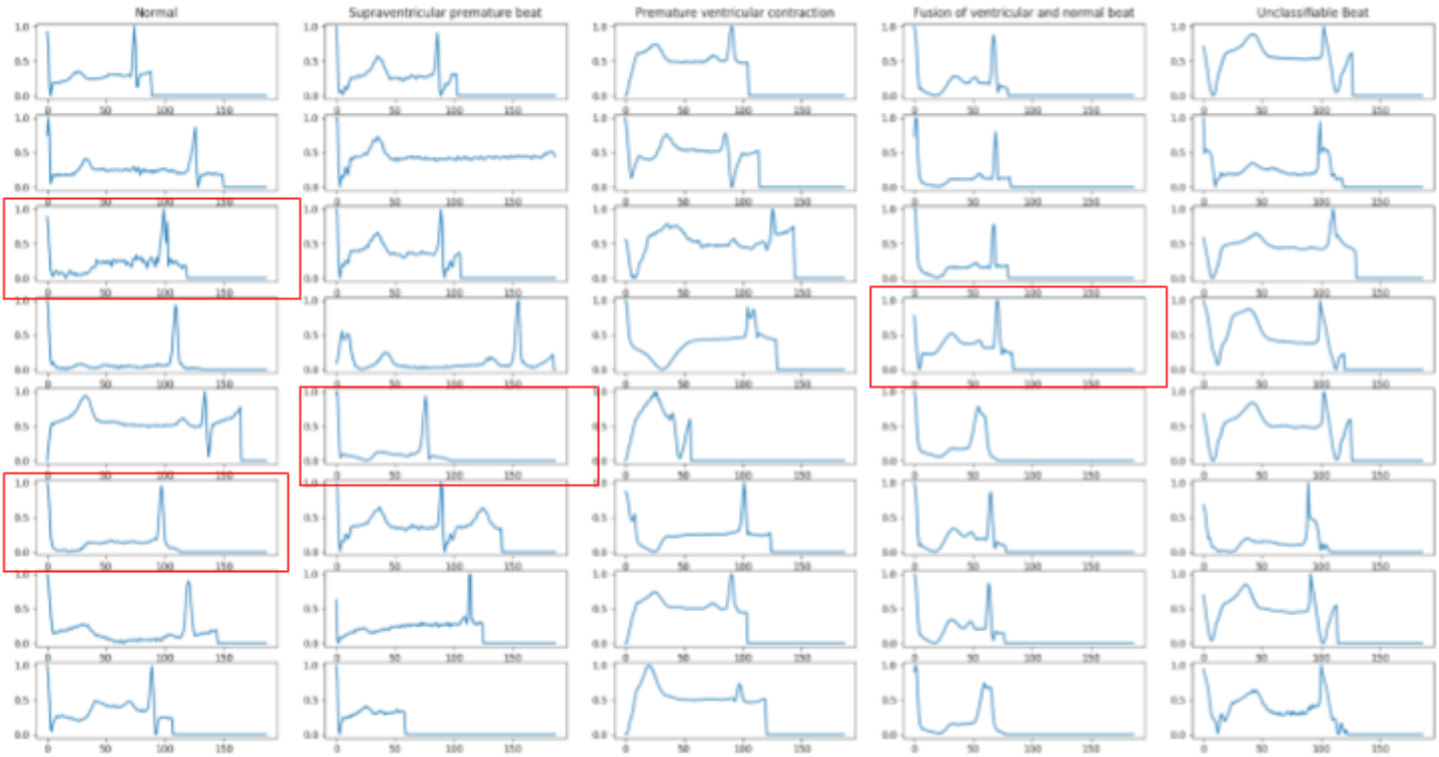
Confusion Matrix for 1 Convolution Pooling 1 Convolution



Confusion matrices for other combinations of convolution and pooling layers can be found in the appendix.

Predicting Heart Beat Classifications

Our network successfully classified heartbeat arrhythmias with an accuracy greater than 94%. In order to better understand the network, we also analyzed a subset of data to visually inspect the heartbeats that are misclassified. In our initial testing subset of 40 heartbeats, four were misclassified by our network. Two of the normal beats were misclassified as unclassifiable beats, and two abnormal beats were mistakenly classified as normal beats. By inspecting the data, these misclassified beats are not visually distinctive to fit within their target categories. The misclassified supraventricular premature beat still has a strong normal beat, and the misclassified fusion is not distinctly abnormal. The first normal that was misclassified is very noisy, so it makes sense that it could have been mistaken as unclassifiable by any observer - human or machine. The second misclassified normal beat a seems more evidently normal, but there may be more subtle data for why the network misclassified it.



Conclusion

We concluded that by using a 1-Dimensional convolutional neural network we can successfully classify heartbeats into one of the following 5 categories: Normal, Supraventricular Premature Beats, Premature Ventricular Contractions, Fusion Ventricular and Normal, and Unclassifiable. We can confidently say that this technology would have a positive impact on the healthcare community and we fully expect to see neural networks and other forms of AI heavily involved in healthcare in the future.

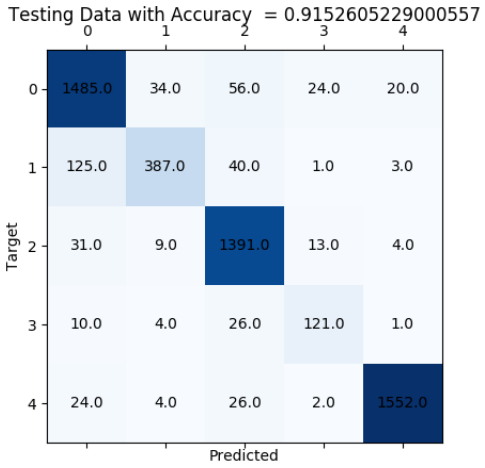
Sources

https://www.kaggle.com/shayanfazeli/heartbeat#mitbih_test.csv

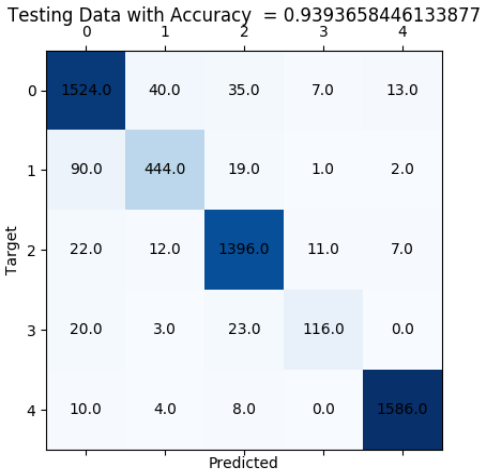
https://physionet.org/physiobank/database/html/mitdbdir/mitdbdir.htm

https://www.geisinger.org/health-and-wellness/wellness-articles/2018/02/09/16/06/4-reasons-behind-an-irregular-heartbeat

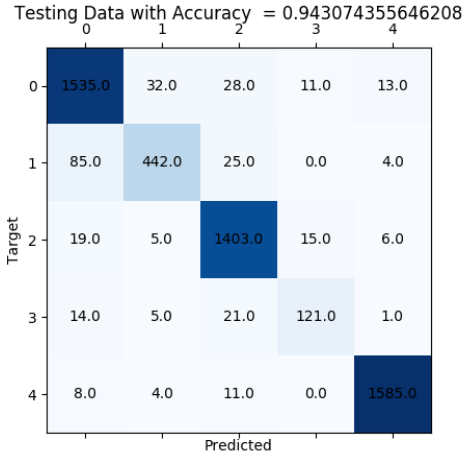
Appendix (Additional Confusion Matrices)



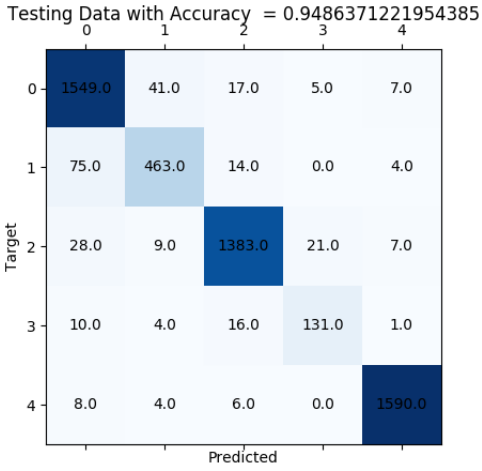
5 epochs



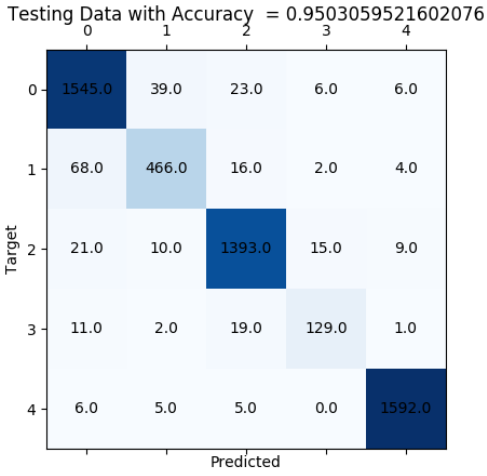
20 epochs



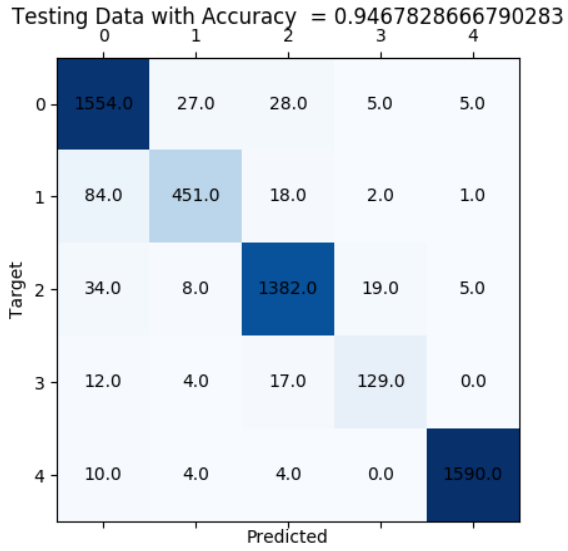
50 epochs



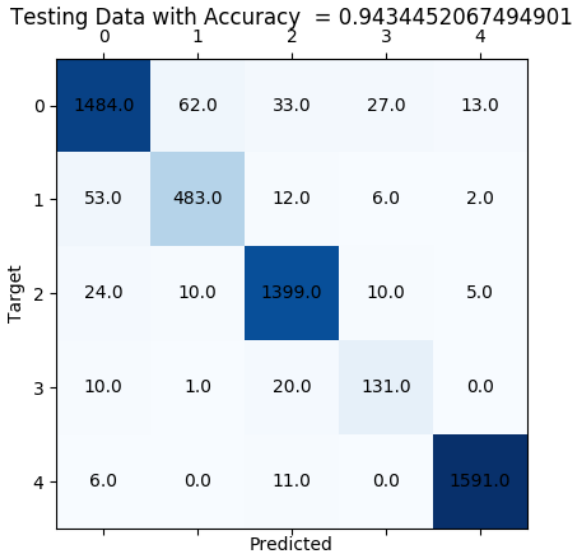
100 epochs



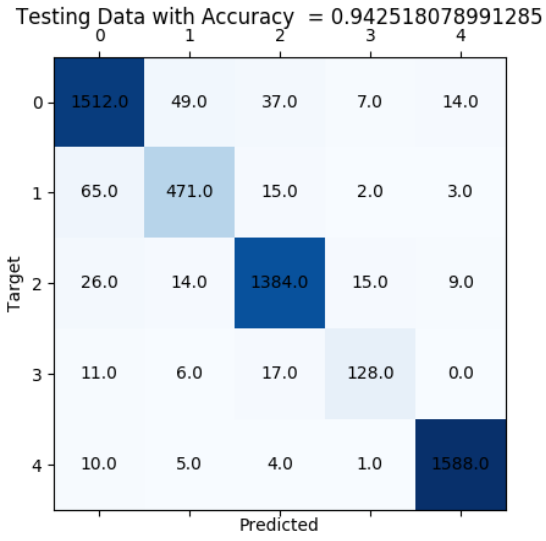
150 epochs



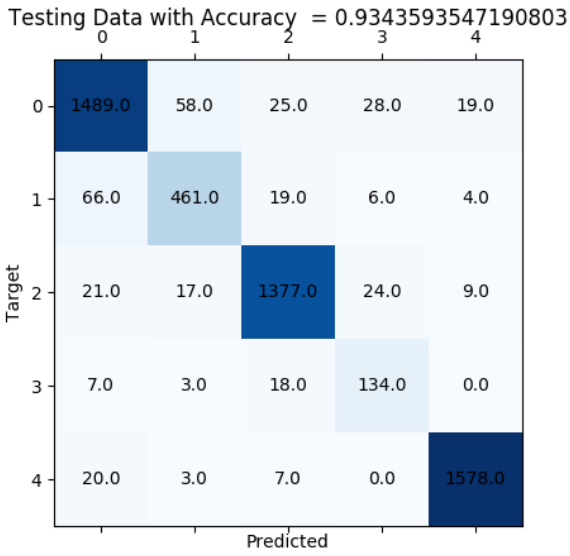
1 Convolution
Pooling
1 Convolution



2 Convolutions
Pooling
2 Convolutions



2 Convolutions
Pooling
2 Convolutions
Pooling



4 Convolutions
Pooling
4 Convolutions